SEVENTH QUARTERLY REPORT

on

AUTOMATIC LANGUAGE ANALYSIS

Indiana University

March 1, 1962

Sponsored by: Air Force Systems Command,
Rome Air Development Center,
Griffis Air Force Base, New York

Contract No.: AF 30(602)-2185

Principal Investigators:
F. W. Householder, Jr.
J. P. Thorne

Linguistics,
Indiana University,
Bloomington, Indiana

In addition to the work reported on here a reverse order dictionary containing some 80,000 entrees has also been produced.

A NEW APPROACH TO THE MECHANICAL ANALYSIS OF ENGLISH

Rita Burns, Ramon Cook, James Peter Thorne

I

Summary

A new approach to the mechanical analysis of English is outlined. Essentially it is a technique of analysis by successive approximation. It uses a more or less conventional dictionary giving form class (part of speech) assignment, or assignments, to each item. With some exceptions, in each case where a word is assigned to more than one form class it enters a routine to determine the class to which it belongs in the particular case under analysis. These routines involve the examination of the immediate environment of the ambiguous item and are based on a close study of group structure in English. It is freely admitted that all such ambiguities cannot be correctly resolved in this way; though a surprisingly large number can. For this reason a strategy was adopted whereby these preliminary routines were designed not merely to produce the greatest number of right solutions but to reduce mistakes as far as possible to those incorrect solutions which can be recognized and corrected at a later stage in the analysis. The routines are constructed in such a way that a wrong assignment at the word level usually means that results will be produced by the subsequent clause analysis routines which are manifestly wrong. All the information concerning each word is preserved so that the most likely candidate for reassignment can be chosen in these cases and the clause routine reapplied. This is called conditional re-entry. It is continued until a legitimate result is produced. Preliminary tests indicate that a very high percentage of correct analyses can be obtained in this way.

A NEW APPROACH TO THE MECHANICAL ANALYSIS OF ENGLISH

Rita Burns, Ramon Cook, James Peter Thorne

I

Summary

A new approach to the mechanical analysis of English is outlined. Essentially it is a technique of analysis by successive approximation. It uses a more or less conventional dictionary giving form class (part of speech) assignment, or assignments, to each item. With some exceptions, in each case where a word is assigned to more than one form class it enters a routine to determine the class to which it belongs in the particular case under analysis. These routines involve the examination of the immediate environment of the ambiguous item and are based on a close study of group structure in English. It is freely admitted that all such ambiguities cannot be correctly resolved in this way; though a surprisingly large number can. For this reason a strategy was adopted whereby these preliminary routines were designed not merely to produce the greatest number of right solutions but to reduce mistakes as far as possible to those incorrect solutions which can be recognized and corrected at a later stage in the analysis. The routines are constructed in such a way that a wrong assignment at the word level usually means that results will be produced by the subsequent clause analysis routines which are manifestly wrong. All the information concerning each word is preserved so that the most likely candidate for reassignment can be chosen in these cases and the clause routine reapplied. This is called conditional re-entry. It is continued until a legitimate result is produced. Preliminary tests indicate that a very high percentage of correct analyses can be obtained in this way.

Introduction

The fact of the pervasiveness of homophony in a language as morphologically poor as English does not need to be laboured: though it is unlikely that anyone who has not read through a few pages of text deliberately looking for this feature has any idea just how widespread it is. It presents a formidable barrier to any preliminary processing of data, whether for purposes of information retrieval or machine translation. It was decided that attention should be concentrated upon those ambiguities involving the possible assignment verb. This meant the construction of five separate sets of rules for the resolution of the ambiguities; noun/verb present tense (point, stage, face etc.) adjective/verb present tense (clean, complete, close etc.) noun/verb past participle (and sometimes present tense also) (cut, set, felt, thought) verb past participle/adjective (fixed, interested, given etc.) and verb present participle/adjective/noun (meaning, using, running etc.). Subsequently it was decided to add a sixth set of rules, in which the ambiguities involved in words of such idiosyncratic distribution as like, except, might, can, will, even, still, well, and a few others, would be resolved. A computer routine for recognizing inflectional affixes was already in operation.[1]

The approach followed derives from an idea put forward by Dr. M.A.K. Halliday[2] and called by him shunting: though Halliday intends this as a general approach to machine translation in all languages and does not specify the corrective potential that can be built into the procedure, particularly when dealing with languages like English. Basic to this procedure is the

---

[1] See J. Lyons, Fifth Quarterly Report on Automatic Language Analysis.

[2] See M.A.K. Halliday, 'Linguistics and Machine Translation', Zeitschrift fur Phonetik und allgemeine Sprachwissenshaft (University of Berlin) (forthcoming).

concept of levels or ranks. Language is regarded as being built up of a hierarchy of units each one forming the elements in the structure of the unit of the level next above it. Thus morphemes are the structural elements of the unit word, words make up groups, groups clauses, and clauses sentences. In the procedure here described no attention was paid to the smallest unit, morpheme; nor was the possibility of a unit still higher than the sentence (paragraph) considered.

Word Ambiguity Routine Construction

It was decided in the first place to see how successful we could be in an attempt to construct diagnostic procedures involving the examination of each word in the sentence in turn, reading from left to right, resolving each ambiguity as it arises. This not only limited us to the investigation of the immediate environment of each word (few rules involve a search of more than three words to the left or right) but also meant that in most cases the significant part of the environment is restricted to the left hand side, since the right so often contains only information that is itself ambiguous. In practice it turned out that these limitations are much less serious than might be supposed. The compensating gain in simplicity in programming is considerable. (A detailed account of the programming of these rules is given below.) The decision to admit only such simple rules was reinforced by the discovery that information from wider environments could be much more efficiently used after division into tentative clauses had been completed.

It is interesting that although many diagnostic features are fairly obvious some quite powerful ones were not recognized until after a great deal of patient investigation. For example, the usefulness of the fact that plural nouns cannot occupy the same place in structure as adjectives (possible exceptions are goods train and brains trust) was not realized at first. The importance of certain sub-classifications (information regarding the sub-

classes of each of the major form classes is also given in the dictionary) particularly countable noun and two object verb also became clearer as the work proceeded. On the other hand it was realized from the very beginning that the distribution of adverbs in English is too haphazard to be of use in this respect, and most rules involve the instruction not to count adverbs as part of the environment.

The most significant development in the rules, however, came after the notion of conditional re-entry had been evolved. A simple illustration of how this affects the word ambiguity rules is afforded by the case where in the resolution of a past participle/adjective ambiguity, the only diagnostic feature in the environment is the occurrence of the word which immediately before the ambiguous item. Here the instructions are to take the item as an adjective; since either this is right, or, if it is wrong, then a clause without a verb will be produced and an error registered. It should perhaps be emphasized that better results could be obtained from these sets of rules alone if these considerations were not taken into account; but the resultant gain by the subsequent correction of errors would, of course, be forfeited.

These routines are now complete and are currently being programmed for the I.B.M. 709 computer.[3] The noun/verb present tense routine was programmed and tested on the I.B.M. 650. Each program contains about a thousand instructions. It is certain that we have not analyzed all the diagnostic environments but it is hoped that we have included all the most productive ones. Presumably some further improvements can be expected from the examination of the results of extended runsof these routines.

Clause Analysis Routines

Work on the construction of routines for tentative clause analysis is well advanced but far from complete. For this purpose a clause is defined

---

[3]These routines were developed by Henrietta Chen, Helga Feider, Patricia Huffman, Tokuichiro Matsuda, and Bruce Moore.

as that part of a sentence containing one and only one verb (modal verbs excepted). Hence the decision to concentrate on verb ambiguities. It has been established that clause analysis is most successfully performed by starting at the end of the sentence and working towards the beginning. The explanation for this seems to rest on the facts that the typical English clause reveals greater complexity in the post-verbal part than the pre-verbal part, and that the beginnings of English clauses are better marked than the ends (if these can be said to be marked at all). Work has been directed towards compiling lists of items at which a break always occurs (a much longer list than can be derived merely by looking at traditional grammar books) and those at which a break can be made if a verb has previously occurred. Rules for deciding where to break when the only evidence that a clause boundary has been crossed is the occurrence of a second verb are also being devised. Here again it is obviously impossible to produce correct analyses by the use of a single procedure. Certain corrective devices have already suggested themselves; notably ones involving counting commas within tentative clauses.

Conditional Re-entry

Work on this phase obviously awaits the completion of the rules for clause analysis, but there seems to be no doubt that the phenomenon we are reporting does regularly occur (which is not to say that the problems of recognizing it when it occurs have been solved). A fairly frequent case is the analysis generated for a one clause sentence in which no element is assigned as a verb. (Unfortunately it has to be admitted that any writer is likely very occasionally to produce a "sentence" without a verb.) But correction by conditional re-entry is far from being limited to one clause sentences or complex sentences where the clauses are defined by overt markers as the following typical example shows:

> One of the most satisfactory laboratory experiments in the field of mechanics/is the measurement of surface tension by means of a DuNouy tensiometer.

The word underlined is wrongly assigned as a verb by the word ambiguity routines resulting in the sentence being analysed into two clauses as indicated. But such a juxtaposition of clauses in English is impossible.

It is particularly encouraging that the quite frequent errors produced when the left hand side of the environment consists of a conjunction such as and or but which can link either two clauses, two groups, or two words, are especially easy to spot at the clause level. In the following sentence:

> It is only at certain times of day and certain times of year/that he in fact succeeds/in observing bands/, and further developments in technique/are therefore required.

the wrong assignment of further produces the incorrect analysis shown, which is recognizable as such not only by the impossible juxtaposition of the last two clauses, but by the change in number without any intervening mark of subordination.

Information concerning the history of the analysis of each word including the number of the rule by which its ambiguous assignment was resolved will be carried forward in the output of the word-ambiguity routines. The rules of each routine are ordered according to their efficiency. In a case where there are two or more candidates for reassignment the word with the lowest rule number is reassigned first.

It will be seen that conditional re-entry is a very simple feedback device. The output of the clause analysis routine itself produces changes in the input (i.e. the results of the word ambiguity routines).

The Dictionary

As the work has progressed it has become clear that certain extensive simplifications can be introduced in the handling of the information supplied by the dictionary. For example any word entered as a preposition can, for

diagnostic purposes, always be treated as if it were only a preposition no matter what other classes it can also belong to. A word like since can always be treated as preposition in the resolution of word ambiguities. The decision as to whether since itself is, in the particular case, a preposition or a conjunction depends on the examination of a much wider context and is postponed until the process of setting up tentative clauses. In fact in most cases a clear-cut distinction can be made between the information that must be obtained from the dictionary and the information that must be obtained by analysis. The fact that in nearly all instances both kinds of information are actually given in the dictionary tends to be merely misleading.

The ambiguity routines work more efficiently if certain phrases, particularly those prepositional phrases such as of course and in fact, which occupy the same place in structure as adverbs are entered in the dictionary as single items. This affords the opportunity for introducing other devices for improving the working of the routines. For example, if the other, others, and other are entered separately, the second can be taken solely as a pronoun and the third solely as an adjective — both potent diagnostic features. Similarly small need only be entered as an adjective since in the phrase the small of the back it occupies a place in structure which can only be filled by nouns. There are numerous other cases.

The last example raises a more general point. The ambiguous classification noun/adjective does not occur in the dictionary at all since this can only ever be resolved by the analysis of structure. This, like the other ambiguities hitherto neglected, adjective/prounoun and adjective/adverb is among the last to be handled — in routines for building up groups within clauses. The final decision regarding the assignment of past participle preceded by part of to have and present participle preceded by part of to be, which up to this point have been treated as adjectives, as either adjectives

or non-finite verbs is also most conveniently made at this point. These are the simplest of all the routines in spite of the fact that ambiguities are resolved and group boundaries established simultaneously. The decision as to which is structurally a noun and which an adjective in a string of what by dictionary look-up or ambiguity routines have been assigned as nouns and adjectives involves little more than counting. This has interesting implications. English has been typologically classified as a langue groupante. Certainly the greatest structural complexity is found at the group level. It is perhaps worth emphasizing therefore that approached in this way group analysis becomes the easiest part of the whole task.

II

Computer Technique for Resolution of Word Ambiguities

Introduction

The computer technique for resolving form class ambiguities revolves about the application of the rules, developed by the linguists, to the words in a sentence. Although the application of these rules is the core of the problem and the main program in the computer technique, it by no means represents the total computer problem. This is essentially a data processing problem. It is a system of many programs which manipulate the text, extracting significant facts and building the files of information which feed the main computer program.

An experiment, conducted on the IBM 650 computer, tested a technique for rule application. Only the set of noun/verb present tense rules was used in this experiment. Input data was simulated for the experiment since programs for information file development were not available. The technique required two computer programs; one composed of approximately 500 instructions;

and the other, of over 1000 instructions. Since the results of the experiment proved satisfactory the same technique has been employed in the main program of the IBM 709 data processing system currently under development.

The main computer program of the IBM 709 system involves the application of the six sets of rules mentioned earlier in this report. Each rule set is being written in subroutine form to be called upon by a major control program. This program, with its subroutines, will be described in detail in a later section. The preparation of this portion of the system is being carried out by five programmers.[4] At the same time these programmers, and a sixth,[5] are writing the auxiliary programs required to build the information files for the main program of the system. All IBM 709 programs are coded in the FAP language[6] and make use of the IOCS[7].

The remainder of this report describes the data processing system for resolving part-of-speech ambiguities. A block diagram of the system is given in Appendix C.

Data Preparation

The raw material used in the computer solution to the word ambiguities problem can be divided into two categories: dictionary data, and scientific text data.

Dictionary Data

The former category was prepared as a tape file for use with the IBM 650 computer programs. This dictionary tape file has been converted by an IBM 7090

---

[4] Rita Burns, Ray Cook, Kay Estes, Jerome Wenker, and S. S. Varma.

[5] Joseph Buckley.

[6] 709/7090 Processing System Bulletin J28-6098-1, 7/61, Fortran Assembly Program (FAP) for the IBM 709/7090.

[7] IBM Reference Manual C28-6100-1, 709/7090 Input/Output Control System.

routine[8] to a format which is suitable for the University's present computers, the IBM 1401 and 709. At the time of its conversion, the dictionary file was modified both in content and in structure to its present format. (See Appendix 3, File 1.) The preparation of this category of data is, therefore, complete. Additions, deletions or changes to the file will hereafter be made through the updating program which will be described in a later section.

Text Data

The initial scientific text, Planet Earth,[9] was also prepared as a tape file for use with the IBM 650 Computer programs and it also has been converted by the above-mentioned routine[8]. (See Appendix B, File 2.) A special IBM 709 program is being written which will be used for assigning to words of this text only the text identification described below.

The preparation of scientific text data is a continuous part of the data processing system developed for this project. A description of the manner in which this data is prepared follows.

Text data preparation is carried out in two stages: the conversion of the printed material to punched cards; and the conversion of the punched cards to tape files.

Text-to-card

The text is punched (See Appendix A, Format 2.) in much the same way as it is typed.[10] The following conventions are observed in the key-punch operation.

---

[8] D. E. Flanigan, IBM, Tape Conversion Routine for the IBM 7090.

[9] Planet Earth. Karl Stumpff (Ann Arbor, 1959).

[10] Titles, subtitles, and questions are not punched.

1) Every sentence begins a new card.

2) The punching format for the text portion of the card is represented by:

F (P_1) W (P_2) S where

F is the format of the word, i.e., upper case, italics, etc.;

P_1 is the punctuation which precedes the word, i.e., quotes, etc.;

W is the word itself as it appears in the printed material[11];

P_2 is the punctuation which follows the word, i.e., comma, dash, quotes, etc.; (P_2 may occupy several card columns.)

S is a blank column which designates the end of the word in text.

3) A sentence may be continuous over many cards. When one-less-than-the-column-limit per card is reached within the middle of a word of the text, a hyphen is inserted and the word is continued on the next card.

4) Each punched card contains a ten-digit identification number. This number serves a dual purpose. It maintains the proper card sequence, and it provides a means of locating the sentence within the text. The identification number assigned in the key-punch operation is divided as follows: digits 1-3 represent the page of text; digit 4 represents the paragraph on the page; digits 5-6 represent the sentence number within the paragraph; and digits 7-10 are zeros. (A page number does not change unless there is also a paragraph change. This means that when a paragraph is continuous from one page to the next, the page number on which the paragraph originates is the page number in the identification number for all sentences within that paragraph.)

Card-to-Tape

The primary purpose of the card-to-tape conversion operation is to provide an efficient form of input to the data processing system. Its

---

[11] Numerals and symbols are punched as NNN and SSS, respectively, in this field.

secondary function is to refine the identification number which was assigned during the key-punch operation.

For the purpose of resolving part-of-speech ambiguities in sentences, certain boundaries are recognized within the ultimate bound of the sentence itself. Some can be readily identified by punctuation marks. Dashes and parentheses, for example, are obvious symbols of breaks within the sentence. The text identification number assigned during the card-to-tape operation makes possible the division of the sentence into its "analyzable units."

The format of the 10-digit text identification number is:

PPP/P SSCUWD where

PPP is the page number in the text;

P is the paragraph number on the page;

SS is the sentence number within the paragraph;

C is the clause within the sentence;

U is the unit within the clause; and

WD is the word in the sentence.

Each word of the text is converted to one logical record on tape (See Appendix B, File 2), and is identified by the above text identification number. By standard IBM sort[12] routines the text file may be sorted in one of two sequences depending on which digits of the text identification number are keys of the sort. In the first example which follows, the keys are digits 1-6 and 9-10. This sequences the text file — one sentence in this case — into original text order. In the second example, the sort keys are digits 1-10. This sequences the file into "analyzable unit" order.

---

[12] IBM Reference Manual C28-6036, 1959, GENERALIZED SORTING PROGRAM FOR THE IBM 709 DATA PROCESSING SYSTEM SORT 709.

	the	boy	I	saw	him	from	the
ID	0011011001	0011011002	0011011103	0011011104	0011011105	0011011106	0011011107

	window	went	to	the	store;	the	girl
ID	0011011108	0011011009	0011011010	0011011011	001101 12	0011012013	0011012014

	went	to	the	movies.
ID	0011012015	0011012016	0011012017	0011012018

Example 1

The boy I saw him from the window went to the store the girl went to the movies.[13]

Example 2

The boy went to the store I saw him from the window the girl went to the movies.[13]

File Standards

For purposes of uniformity within the data processing system, all files conform to the following standards:

1) Tapes are written at low density;

2) Logical record length within any file is fixed;

3) Block (physical tape record) length within any file is fixed;

4) Data files are all BCD files;

5) In addition to its data blocks, each data file contains one header record and one trailer record as described in the IOCS manual.[14]

6) Each blank tape mounted for use as output of the 709 computer programs contains the Blank Tape Label described in the IOCS Manual.[15]

File Maintenance

The data processing system for resolving word ambiguities requires two types of maintenance programs: sort programs and update programs.

---

[13]Since the punctuation which determines the units is no longer needed for resolution of word ambiguities, it is omitted from the tape record at card-to-tape conversion. The text identification number may be used to reconstruct this punctuation if it is later considered important. All other sentence punctuation is converted to tape.

[14]IBM Reference Manual C28-6100-1, 709/7090 INPUT/OUTPUT CONTROL SYSTEM, pp. 22,23.

[15]Ibid., p. 21.

Sorting is carried out by means of the standard IBM 709 Sort[12]. Sort sequences are designated both in the file descriptions (Appendix B) and in the block diagrams (Appendix C).

Updating of all files of the system may be carried out by means of a single IBM 709 program which is currently being written. Although intended originally for updating the dictionary file only, this program has been designed as a multi-purpose routine for updating both data and instruction files. In addition to enabling changes to be readily introduced to these tape files, the program provides a means of examining selected parts of the files' contents.

Update Program

The particular function which the program performs at any one time depends upon the specifications stated by the user. These specifications include a general description of the characteristics of the file to be modified and of the manner of modification (Appendix A, Formats 3 & 5), and a detailed description of each type of modification (Appendix A, Formats 4 & 6).

Specifications are punched on cards and converted to tape by a standard IBM 1401 program in which each card becomes one fourteen-word record. (See Appendix B, File 3). This "change" tape and the tape file to be modified are the two inputs to the file update program. The primary output of the program is the modified file. A secondary output file can be produced if requested in the specifications.

Instruction Files

Instruction files are updated by means of the specification cards #5 and #6. These cards direct the program to locate a block by its position within the file and to change a word within that block. There may be any

number of alterations within the same file block, each specified by a distinct card #6.[16]

Data Files

Data files are updated by means of cards #3 and #4. Depending on the use of the parameters in these cards the program may be directed to:

1) locate a logical record(block) by its position in the file, and add N logical records(blocks) immediately after the one located.

The actual records(blocks) added follow immediately behind the specification card #4. Each of the N records(blocks) must begin in column 1 of a card; the record(block) may be continued from card to card (78 columns per card) until the full size of the record(block) has been punched. (Continuation cards are identified by a "c" in column 79.) Final ∅ padding of records(blocks) need not be punched. Since the record and block size are both stated in card #3, a full record(block) will always be added.

2) locate a record(block) by its position in the file, and delete — beginning with that record(block) or the following one — N records(blocks). No records(blocks) need follow this specification card #4.[16]

It is possible in updating data files to delete certain records(blocks) and to add others at the same point in the file. This must be done by means of two specification cards #4, each referring to the record(block) after which records(blocks) are to be added. The specification card with the "Add" parameter and the records(blocks) to be added must precede the specification card with the "Delete" parameter.

---

[16] By this direction each specification card #4 (6) applies to a single record(block) in the file. Once this record(block) is located and modification made the specification card has no further application. This type of direction assumes the "change" file to be in the same order as the file to be modified.

3) locate[17] a record(block) by its position, and change that record(block) of the file with the match field stated on the specification card #4. When a record(block) is found in which the two fields agree, delete or add N records(blocks), or change that record(block) by the replacement field of the specification card #4.[16]

5) locate[17] a specified field in every record(block) of the file. Whenever the field specified matches the field stated on the specification card #4, change the record(block) by the replacement field of the specification card #4.[18]

The dictionary file (See Appendix B, File 1) may be updated by record in any of the above-mentioned ways. However, in addition, the logical records within the dictionary file which contain the count of the number of other records within an alphabetic grouping are modified in accordance with the addition or deletion of records during updating.

Data files may also be examined by means of the specification cards #3 and #4. When used for this purpose, the specification cards direct the program to locate a specified field in every record(block) of the file. Whenever the record(block) field matches the field specified in card #4, the record(block) is placed on the secondary output file.

---

[17] When the locating is carried out at the record level neither the matching field nor the replacing field may exceed the size of the record. When the locating is carried out at the block level(block) neither the matching field nor the replacing field may exceed the size of the block. The match field must always be completely stated on the specification card #4; the replacement field must begin on the specification card but may continue from card to card until it is completely stated. No characters — including zeros — may be omitted from the field statement.

[18] By this direction all specification cards #4 are applied to each record(block) of the file. This makes possible a change which is universal to all records(blocks) of the file, or, if no change is requested, the selection of all records(blocks) which have some common characteristic.

All specification cards #4 except those using the "Add" parameter may request secondary output. When requested in connection with deletions, the secondary output file will contain all records(blocks) which have been deleted from the modified file.

When requested in connection with changes, the secondary output file will contain all records which have been modified on the primary output file.

Information Gathering

With data files prepared, updated and maintained in various sort sequences, the next stage in the data processing system is one of information gathering. A single program, the Affix-Dictionary program, has been written to gather information from the dictionary file and append it to the text file.

The Affix-Dictionary Program of the IBM 709 system is quite different from the program written for the IBM 650 Computer.[19] The difference stems from the development of the project's aims in the last few months. The initial program was intended to provide the linguists with a step-by-step picture of form class derivation by affix removal. The Affix-Dictionary program of the 709 system was designed specifically to simplify the task of resolving word ambiguities. For this reason the 709 program produces only the resultant form classes and that information regarding them which contributes significantly to ambiguity resolution. In addition to the basic change in purpose between the two programs there are changes based on the new format and content of the dictionary file described earlier in this report.

Affix-Dictionary Program

Input files to the Affix-Dictionary Program are two: the dictionary file (Appendix B, File #1) and the text file (Appendix B, File #2). The

---

[19] Fifth Quarterly Report on Automatic Language Analysis

latter is in alphabetic word sequence to enable a straightforward matching of the two files.

Output from the program is an appended text file (Appendix B, File #4) and an Error File (Appendix B, File #5). The Error File is produced when there is no matching dictionary entry for a word of the text or for its stem after affix removal. When this situation arises, additional entries must be made to the dictionary file by means of the Update Program. The Affix-Dictionary Program must then be repeated with the updated dictionary file as input.

By means of this program the 10-word record (Appendix B, File #2) representing a word of text increases to a 20-word record. The Affix-Dictionary Program inserts into the expanded record (Appendix B, File #4) codes representing all the form classes to which the word of text may belong. A code indicating a preferred form class, if there is one, is also added. Subclassifications of form classes and the stem (word found in the dictionary) are inserted. Coded representations of those affixes removed (prior to locating the stem) and the next removeable affix are included in the expanded record.

Although more information must be known about each text word before ambiguities can be resolved, the remaining information can be gathered in the Resolution of Ambiguities program which will be described in the next section.

Once the appended file has been produced it is sequenced into the analyzable unit order (Text Identification) in order that the ambiguities may be resolved sequentially from left to right.

Resolution of Word Ambiguities

The resolution of word ambiguities within an analyzable unit is carried out on the IBM 709 Computer in a manner which imitates the method employed

by the linguist. The procedure in its simplest form might be stated as follows:

1) Within an analyzable unit (left to right), note all words which are members of only one form class, i.e., never ambiguous;

2) Within the same unit (left to right), resolve the ambiguities of certain words of unusual distribution, as mine or like; (This step involves the application of one of a group of rules designed specifically for the resolution of ambiguities of words in this class. Examination of the immediate environment of the words is required.)

3) Within the same unit (left to right), determine the particular type of ambiguity, such as noun/verb present tense or noun/verb past tense. (The ambiguity is resolved by applying a suitable rule from a group designed for the resolution of this type of ambiguity. The application of the rule requires testing the immediate environment of the ambiguous word.)

The program written to follow this procedure consists of a control routine (See Appendix D, Control Program) and a group of subroutines.

Control Routine

The control routine reads the analyzable unit from the appended text file into the computer memory, noting (step 1 above), as it reads, the words which are unambiguous. For each of these words the control routine places the appropriate English word, i.e., NOUN, into a specified part of its memory record.

When the unit has been completely read into the memory, the control routine begins its second pass. In this pass it determines the words belonging to the class defined as "words of unusual distribution" and transfers to a subroutine which applies rules sequentially until it resolves the ambiguity of the word. The subroutine then supplies the English word

for the form class and the resolving rule number to the specified part of the memory record for this word of text. Control returns again to the control routine which proceeds in this fashion until it again reaches the end of the unit.

When all such words have been resolved, the control routine makes its third and final pass through the unit. In this pass it determines the particular type of ambiguity of all other ambiguous words and transfers to the appropriate subroutines for resolving them. These subroutines function in the manner described in the preceding paragraph.

When all words of the unit have been resolved, the control program writes the unit whose records (Appendix B, File #6) now show the resolved form class and the resolving rule number.

The procedure is repeated until all the units of the text have been analyzed.

Indicator Subroutine

Omitted from the above description of the control routine is reference to a special subroutine, the Indicator Subroutine. (Appendix F, Indicator Flow Chart). This subroutine completes the information gathering process begun in the Affix-Dictionary program and provides the control program and the remaining subroutines with sufficient facts for determining the type of ambiguity and for applying the specific rules for its resolution.

Resolution of form-class ambiguities depends upon the analysis of regularly recurring environments (indicator situations). First these indicator situations were broken down into parts; characteristics of the ambiguous item itself (for example, being the first word in the sentence, capitalization, etc.), characteristics of immediately preceding words, characteristics of preceding word + 1, characteristics of preceding word

ignoring non-prepositional adverbs, and so on. Indicator categories were then set up and codes were given to each of these, indicator codes, which could be used in the machine. For example, one part of the indicator situation for several rules among the six rule sets is the presence of a modal, copulative, or auxiliary verb. For one situation a member of this class is required to be immediately in front of the ambiguous item; for another, non-prepositional adverbs may be ignored. Again, for others, a member of this class must immediately follow the ambiguous item, in one case immediately following a word belonging to another category which itself follows the ambiguous item; in another non-prepositional adverbs may be ignored. Finally in one rule it is only required to be the next verb. Nevertheless there is only one indicator code for this class - a "1" in the 10th position of the first indicator word. This means that each word in the sentence is tested to see if it is a member of this class, and if it is, a "1" is placed in the arbitrarily determined position of the arbitrarily chosen indicator word; otherwise there is a "0" in this position. Since this class is a subclass of a slightly larger indicator category a "1" would be put in the predetermined place indicating that the word is a member of this larger class also. Since both modals and auxiliaries are themselves members of other indicator categories, "indicators" must be placed in several places for such an item. Before any attempt is made to resolve the verbal ambiguities every word in the sentence must be tested to see if it fits one of these indicator classes.

On first consideration the task, performed by the control routine, of determining the type of ambiguity of a word seems relatively straightforward. It can be shown, however, to be quite complex, involving many computer instructions. For this reason the control program transfers to the Indicator Subroutine. The Indicator Subroutine performs the necessary tests and classifies each word by inserting a number in the Ambiguous Word Code

(Appendix B, File #6). The control routine need then make only a single bit test to determine which type of ambiguity exists and, thereby, determine which subroutine must be entered.

In providing sufficient information for the subroutines to operate efficiently, the Indicat r Subroutine is even more valuable. An illustration of a rule from one of the subroutines will serve to illustrate its value.

Has it got to immediately in front of it? If yes, see if the ambiguous item has affix "s". If yes, take as a noun. If no see if the preceding verb is one of the group of words belong, attribute, pertain, cling, attach, convert, ascribe, reconvert, commit, oppose, relate, subject, lead, or if the word immediately before to is the same as the ambiguous item, or if this word is one of the group of words similar, propositional, attributable, as, subject, due, prior, resistant, antagonistic, foreign, alien, regard, respect, contrary, liability, inimical, hostile, complementary, equivalent, opposite, amenable. If yes, take as noun. If no, take as verb.

Obviously the rule is complex even in the number of questions it asks before a resolution can be made. Each of the subroutines has between 20 and 40 rules of varying complexity. The additional complication of determining, for example, if the word in question is a member of one of the groups mentioned above increases the rule's complexity from a programming standpoint and decreases its flexibility.

Flexibility is a most important feature of this entire data processing system. It is especially important in the program for resolving ambiguities. There have been changes both in rules and rule ordering. There have also been insertions, deletions, and changes in word groupings such as the groups underlined in the above illustration. It is anticipated that when the results of the computer programs are studied, more changes are inevitable. A decision was made, therefore, that in order to maintain maximum flexibility the rules performed by the subroutines should be stated simply and directly in terms of computer instructions and that the determination of "belonging" to classes should be separated from the rules.

In the example above, the Indicator Subroutine "indicates" that a word of the unit does or does not belong to the group of words — give, etc. by

simply storing a "1" or "0", respectively, in a certain position of computer word in the memory record. The "1" and "0" are known as indicator codes, the computer word in the memory record is known as indicator code word. There are 95 indicator codes. The groups which can be classified by means of indicator codes are varied. Among them are the following:

1) all words which are either modal verbs, copulative verbs, verbs which are forms of to be or to have.

2) all words which are either possessive adjectives, or ambiguous adjective/pronouns, or ambiguous nouns.

3) all words which are either verbs or prepositions or conjunctions or words which end in -ing.

The control routine calls upon the Indicator Subroutine at several different times. By so doing, the information recorded in the indicator code words of the memory records is maintained in its most precise form for use by the subroutines. Instead of testing a word against a long list, for example, to determine whether or not a rule is suitable for resolving a word's ambiguity, the subroutine need test only a single bit position of an indicator code word.

Other Subroutines

It is by means of the subroutines that the computer resolves the ambiguous words of the text. There are six subroutines corresponding to the six types of ambiguities which are to be resolved. Each subroutine is made up of a) a control program, b) a rule table, and c) coding which represents, in computer terms, each of the rules in the set for resolving the specific ambiguity.

The control program is standard for all subroutines. It may be stated as follows:

1) Advance a rule counter C.

2) Locate from a rule table the starting address of the coding, corresponding to Rule C.

3) Transfer to the coding for Rule C.

The coding for Rule C either resolves the ambiguity or returns to step 1) above. If the ambiguity is resolved, the resolved part of speech and the resolving rule number C is supplied to the specified portion of the text word in memory. The counter is reset to ø, and the subroutine returns to the main Control Routine.

The rule table consists of a list of symbolic addresses corresponding to the entry points to coding for each rule in the set. The subroutine executes the rules by transferring in sequence to the symbolic addresses in this table.

The use of a rule table adds another feature of flexibility to the system. By rearranging the sequence of the symbolic addresses in the table it is possible to rearrange the order in which the rules of the set are applied. It is further possible to eliminate the application of certain rules by simply omitting their symbolic addresses from the table. Rules may also be added by supplying the necessary coding for the rule, assigning it a symbolic address, and inserting this address in the desired place in the table. These changes are made by reassembling the program. The rule number is not actually attached to any piece of coding, but corresponds to the order of rule application.

The coding for a rule in the set may require the examination of the indicator codes, the form classes, and other characteristics of the ambiguous word or of the words which precede or follow it. When examination of words other than the ambiguous word is required, the subroutine gains access to these words by means of special "search" routines.

The search routines were developed to meet the common need of all the subroutines. There are eight search routines: four to locate words to the left of the given word and four to locate words to the right of this word.

They differ in their manner of searching. One of the routines, for example, locates the word preceding the given word ignoring all "adverbs & prepositions" which precede it.

The flow chart of a typical rule in Appendix E illustrates the computer technique for applying one of the linguistic rules for resolving noun/verb present tense ambiguities.

Clause Resolution

Although there has been no programming in progress for this part of the system, the general procedure is shown in the block diagram in Block 4, Appendix C.

APPENDIX A — Card Formats

FORMAT #	FORMAT NAME	COLUMNS	FIELD NAME	PARAMETERS
1	Dictionary	1-15	Dictionary Code	
		27	Special Code	
		35-39	Dictionary Order #	
		43-67	Alphabetic Word	
2	Text	1-10	Identification #	
		43-72	Words of Text (Described in Data Preparation)	
3	General BCD Update Specifications	1	BCD Identification	"X"
		2	Dictionary Update Code	D=yes Ø=no
		3-6	Header Block Size	
		9-12	Data Block Size	1=by Data Block#
		15-18	Logical Record Size	2=by Record#
		19	General Update Code	3=by Word in Block 4=by Word in Record
4	Detail BCD Update Specifications	1	Modification Code	A=Add C=Change D=Delete S=Search
		2	Secondary Output	S=yes Ø=no
		3	Starting Block or Record ("D" Cards)	O=same N=next
		5-6	Number of Blocks or Records ("A" and "D" Cards)	01...10
		7-12	or Block or Record# Starting Word in Block or Record of Match Field	
		13	or Ø Starting Digit of Match Field	
		17-18	or Ø Length of Match Field (Digits)≤36	
		19-54	or Ø Match Field	
		26...56-30...60	or Ø Starting Word in Block or Record of Change Field	

FORMAT #	FORMAT NAME	COLUMNS	FIELD NAME	PARAMETERS
		25...55	or { Ø / Starting Digit of Change Field	
		31...61-36...66	or { Ø / Length of Change Field in Digits	
		37...67-78	or { Ø / Change Field (Change Field May Continue in Next Cards Cols. 1-78)	
5	General Binary Update Specification	1	Binary General Identification	"X"
		3-6	Header Block Size	
		7-12	Other Block Size	
6	Detail Binary Update Specification	1	Binary Detail Identification	"Z"
		3-6	Block # of Change	
		7-12	Starting Wd in Block for Change	
		17-18	Starting Bit in Word for Change (1→r=1-36)	
		23-24	Length of Change Field in Bits≤36	
		25-60	Change	

N.B. All card cols. not specified above = Ø. all padding = Ø

APPENDIX B

FILE #	FILE NAME	CHARACTERISTICS	RECORD FORMAT#	WORD	DIGIT	FIELD NAME
1	Dictionary	Block=500 Wd.	1	1	1-	Alphabetic Wd. (Ø Padding)
		Record=20 Wd.				
		Header/Trailer		5	1	
		Sequence:				
		1=Alphabetic Wd.		5	2-6	Dictionary Order #
				6	1-	Dictionary Code
				8	3	
				10	6	Special Code
			2	4	1-4	Prior Alphabetic Group Code
				5	1-2	Succeeding Alphabetic Group Code
				9	3-	Prior Alphabetic Group Record Count
				10	6	
2	Text	Block=500 Wd		1	1-	Alphabetic Wd. (Ø Padding)
		Record=10Wd				
		Header/Trailer		5	1	
		Sequences:				P_2: Post Punctuation
		1 Text Identification		5	2	
		Wd 6, dig 1-6;				P_1: Preceding
		Wd 7, dig 3-4		5	3-6	Punctuation
		2 Text Identification		6	1-	Text Identification
		3 Alphabetic Word		7	4	

N.B. Ø Padding fills all unnamed digit positions of records.

APPENDIX B (Cont'd)

FILE #	FILE NAME	CHARACTERISTICS	RECORD FORMAT#	WORD	DIGIT	FIELD NAME	PARAMETERS
3	Change	Block=14 Wd Record=14 Wd Header/Trailer Sequence: Record 1	1	1	1	BCD Identification	= "X"
				1	2	Dictionary Update Code	D=yes ø=no
		2		1	3-6	Header Block Size	
		•					
		•		2	3-6	Data Block Size	
		•					
		or Record 3		3	1	General Update Code	1=by data block # 2=by record # 3=by word in block 4=by word in record
		4					
		•					
		•					
		•					
			2	1	1	Modification Code	A=Add C=Change D=Delete S=Search
				1	2	Secondary Output	S=yes ø=no
				1	3	Starting Block or Record ("D" Cards)	C=same N=next
				1	5-6	Number of Blocks or Records ("A" and "D" Cards)	01...10
				2	1-6	Block or Record # Starting Word in Block or Record of Match Field	
				3	1 or	ø Starting Digit of Match Field	
				3	5-6 or	ø Length of Match Field (Digits)	ø36
				4 thru 9	1-6	ø or Match Field	

APPENDIX B (Cont'd)

FILE#	FILE NAME	CHARACTERISTICS	RECORD FORMAT#	WORD	DIGIT	FIELD NAME	PARAMETERS
				5...10	2-6	or Ø / Starting Word in Block or Record of Change Field	
				5...10	1	or Ø / Starting Digit of Change Field	
				6...11	1-6	or Ø / Length of Change Field in Digits	
				7...12	1-	or Ø / Change Field (Change Field May Continue in Next Cards cols.1-78)	
				13	6		
			3	1	1	Binary General Identification	"X"
				1	3-6	Header Block Size	
				2	1-6	Other Block Size	
			4	1	1	Binary Detail Identification	"Z"
				1	3-6	Block # of Change	
				2	1-6	Starting Wd in Block For Change	
				3	5-6	Starting Bit in Word for Change (1→r=1-36)	
				4	5-6	Length of Change Field in Bits≤36	
				5	1-	Change	
				10	6		
4	Appended Text	Block=500 Wd Record=20 Wd Header/Trailer Sequence: 1 Alphabetic Wd. 2 Text Identification	1	1	1-	Alphabetic Word	
				5	1		
				5	2	P_2: Post Punctuation	
				5	5	Special Code	
				5	6	Capital Letter Code	

APPENDIX B (Cont'd)

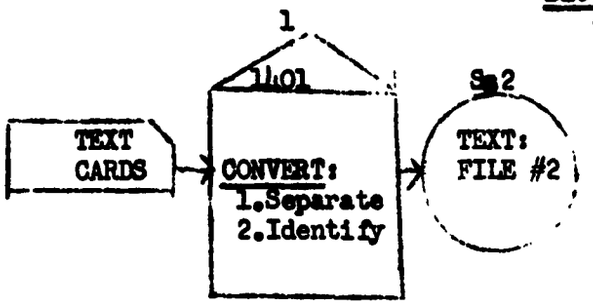
FILE#	FILE NAME	CHARACTERISTICS	RECORD FORMAT#	WORD	DIGIT	FIELD NAME	PARAMETERS
				6	1-	Text Identification	
				7	4		
				7	5-6	Final Affix Code	
				8	1-2	(Final -1) Affix Code	
				9	1-	Alphabetic Word Stem	
				11	6		
				12	1	Noun/Pronoun Code	1=Noun 2=Pronoun 0=Neither
					2	Verb/Conj. Code	1=Noun 2=Pronoun 0=Neither
					3	Adj/Prep Code	1=Noun 2=Pronoun 0=Neither
					4	Adv. Code	1=Adv. 0=Adv.
				12	5-6	Noun/Pron.Subclass	
				13	1-2	Verb/Conj. Subclass	
				13	3	Adj/Prep Subclass	
				13	4	Adv. Subclass	
				20	1-5	Dictionary Code (First Five Digits)	
5	Error	Block=20Wd. Record=10 Wd. Header/Trailer	1	= Record Format 1, File #2			
6	Resolved Text (N)	Block=500 Wd Record=20 Wd Header/Trailer Sequence: 1 Text Identification	1	All Fields Are Identical to Record Format 1, File 4 Additional Fields Are			
				15	1-	Indicator Codes	
				17	6		
				18	1-6	Resolved Part of Speech	
				19	1-6	Resolving Rule #	

APPENDIX B (Cont'd)

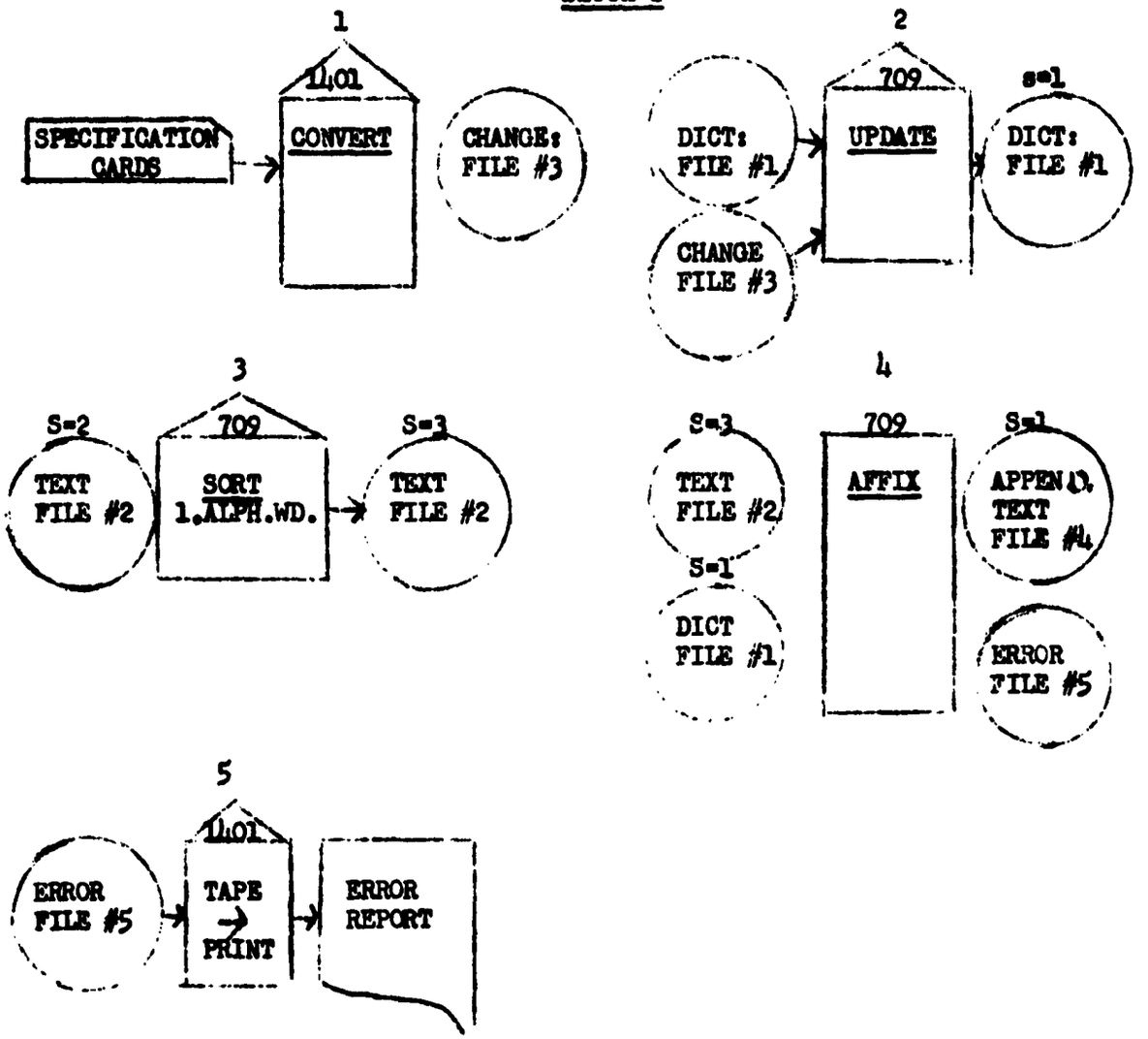
FILE#	FILE NAME	CHARACTERISTICS	RECORD FORMAT#	WORD	DIGIT	FIELD NAME	PARAMETERS
				20	6	Ambiguous Word Code	1=N/V Pres Tense 2=ing 4=adj/past tense 8=adj/pres. tense 16=n/past tense 32=unusual distribution

APPENDIX C

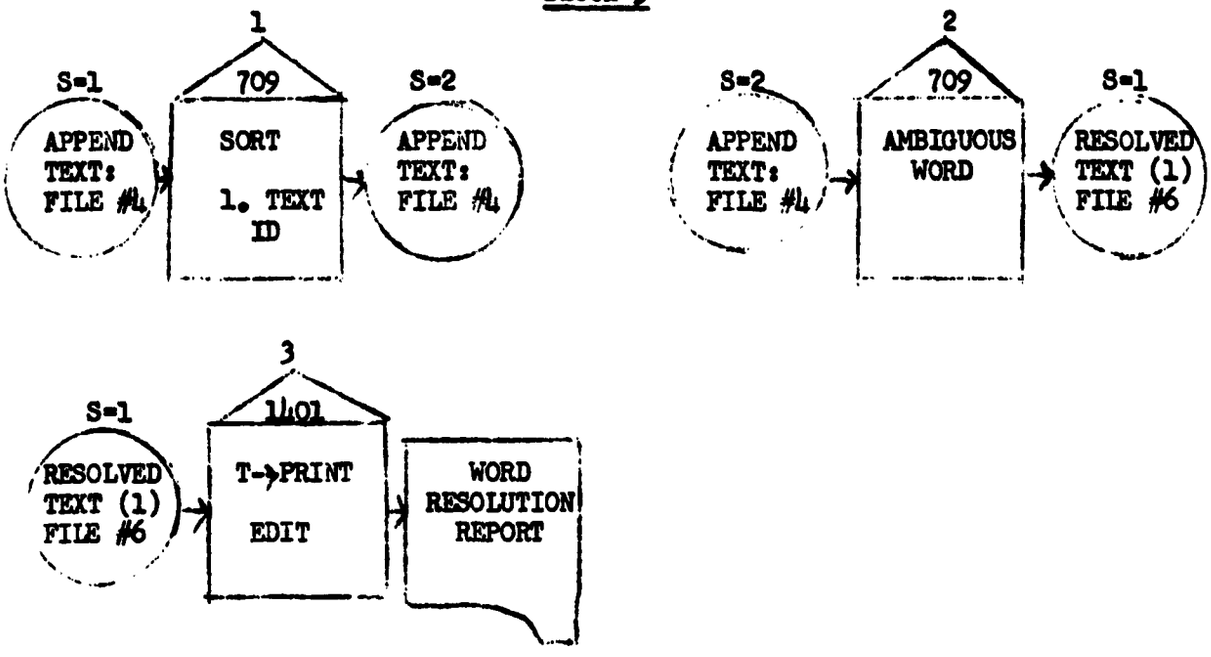
BLOCK 1



BLOCK 2

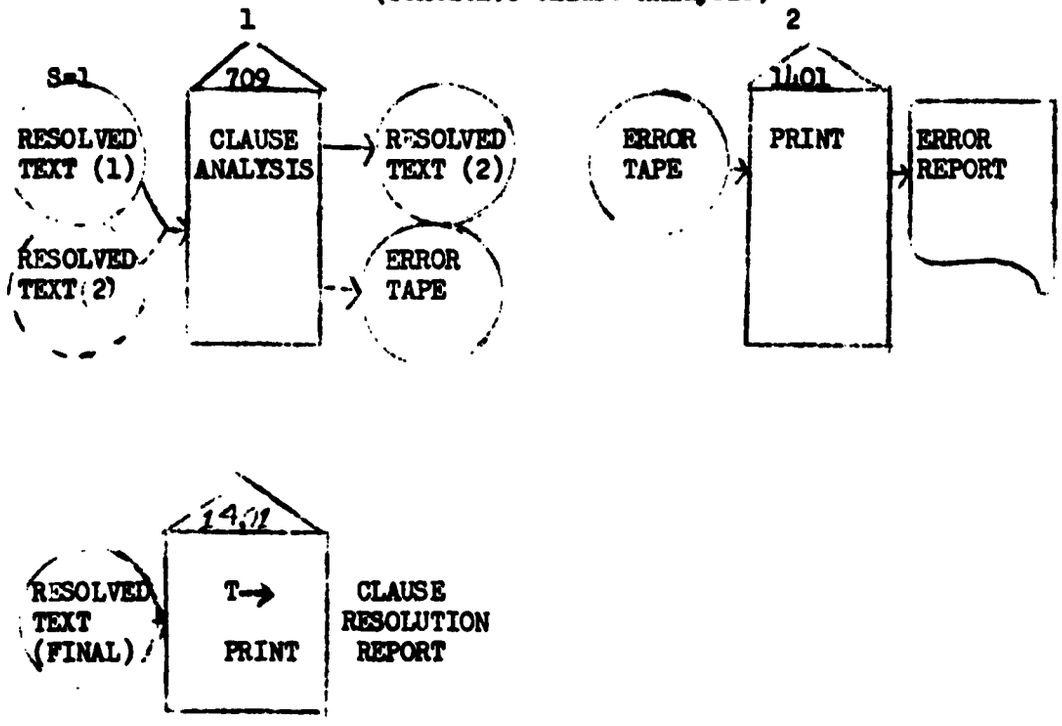


BLOCK 3

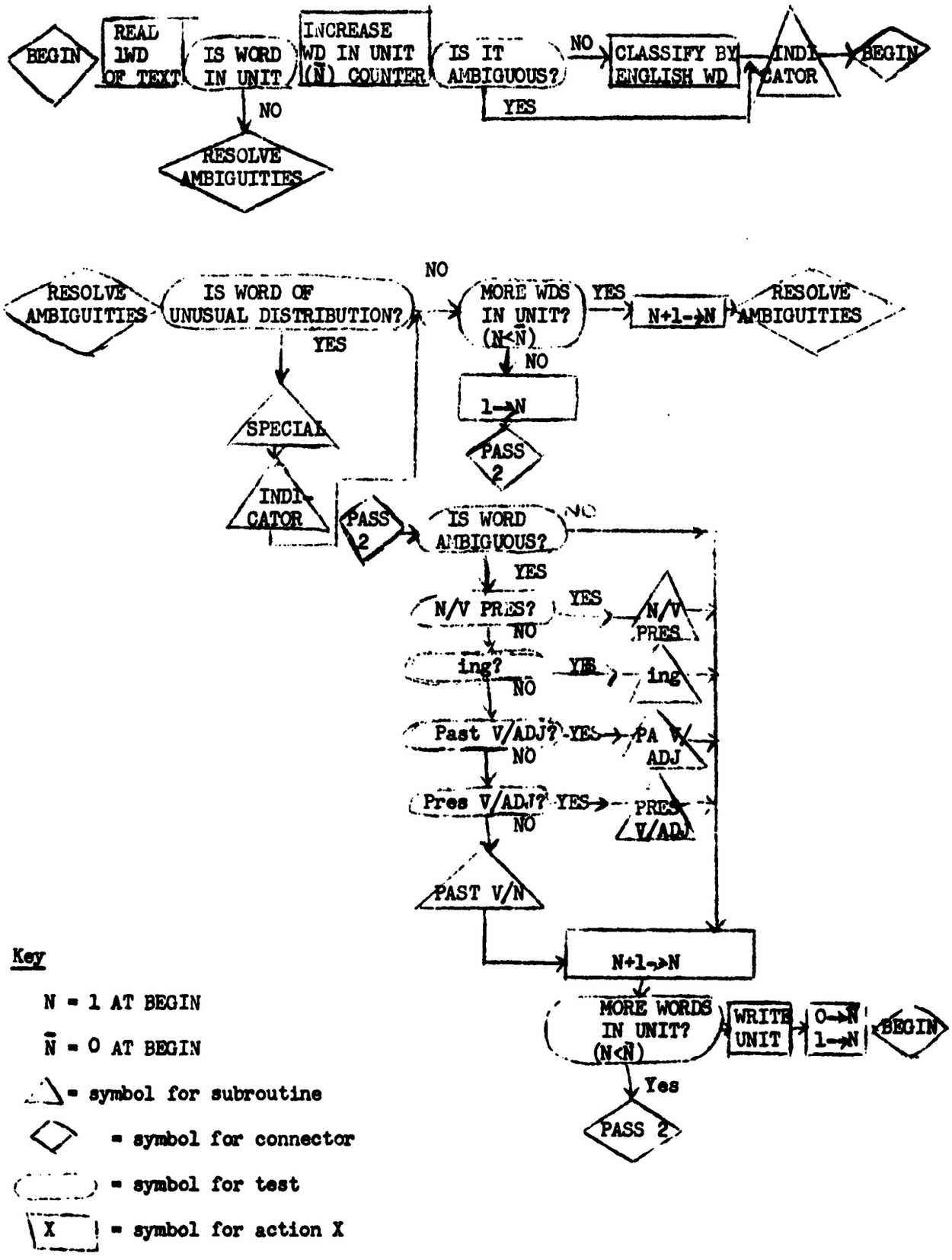


BLOCK 4

(Tentative Clause Analysis)



APPENDIX D Control Flow Chart



Key

N = 1 AT BEGIN

$\tilde{N}$ = 0 AT BEGIN

= symbol for subroutine

= symbol for connector

= symbol for test

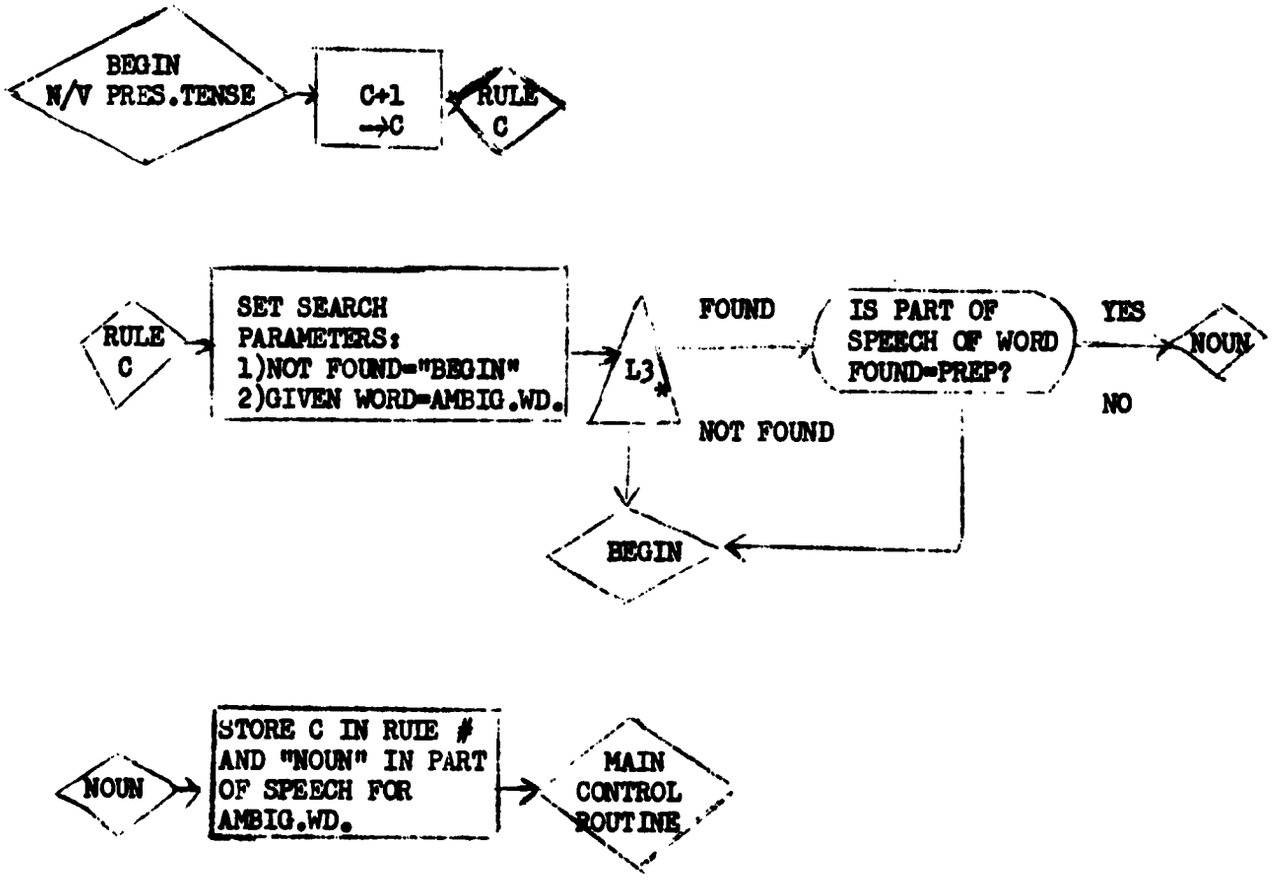
X = symbol for action X

APPENDIX E

Typical Rule noun/verb present tense

Rule C

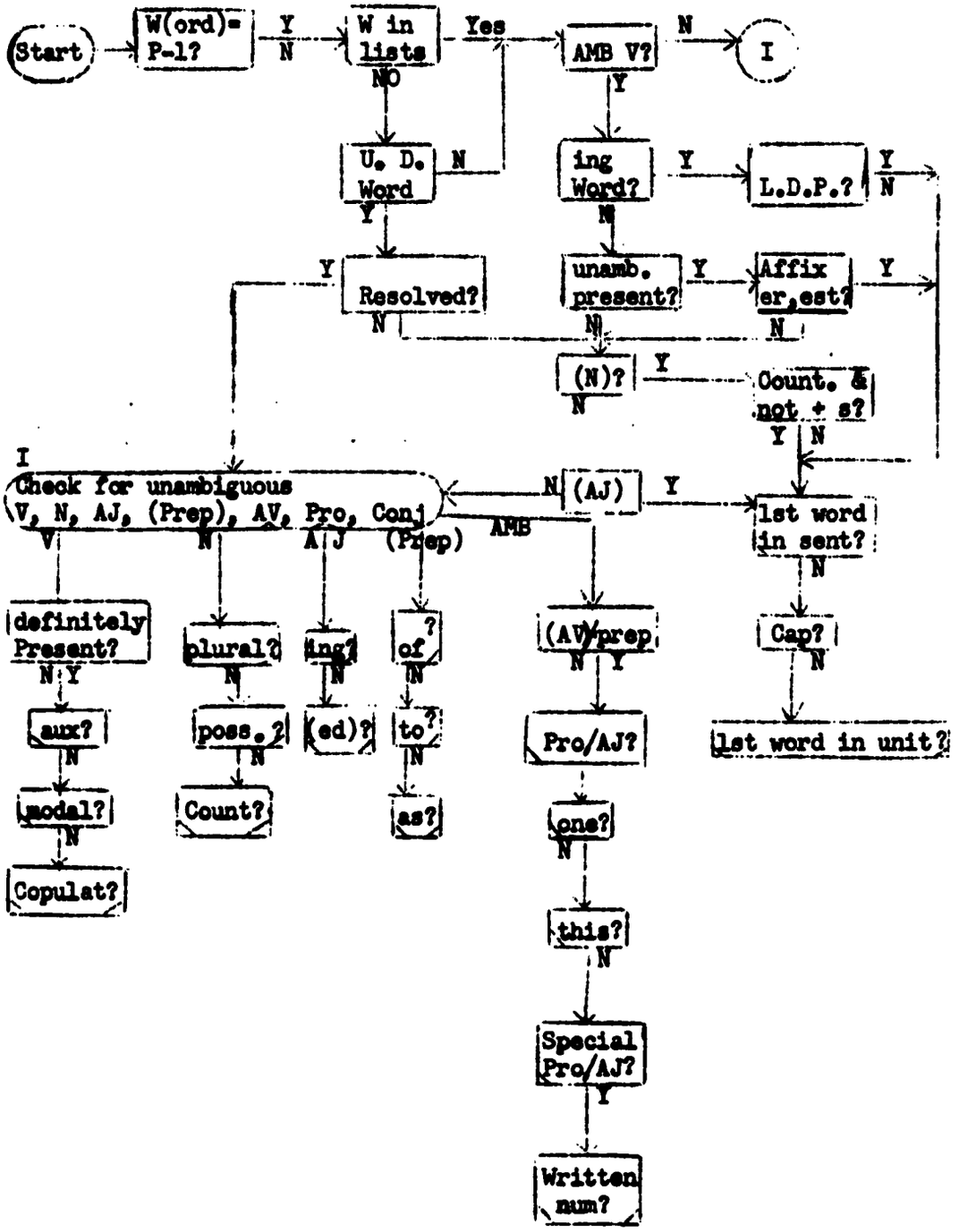
Is the ambiguous word preceded (ignoring adv/ prep and adj/pronouns) by a preposition? If yes assign as a noun. If no, or if no word precedes, apply next sequential rule.



* L3 is a search routine which locates the first word (which does not belong to the class "adv/prep and adj/pron") preceding the given one. If a comma or no word is found to precede the given word, the search routine returns to BEGIN. Otherwise, the memory address of the word found is made available to the subroutine for testing purposes.

APPENDIX F

Indicator Flow Chart



(X) = possibly class X
(ed) = past part. as verb form
Count. = countable
Pro/Aj = exactly Pro-Aj ambiguous
L.D.P. = Limited dist. participle
U.D. = unusual distribution


means one arrow returns to control program


means (having placed indicators) return to control

Of course, each box normally includes the setting of indicators.